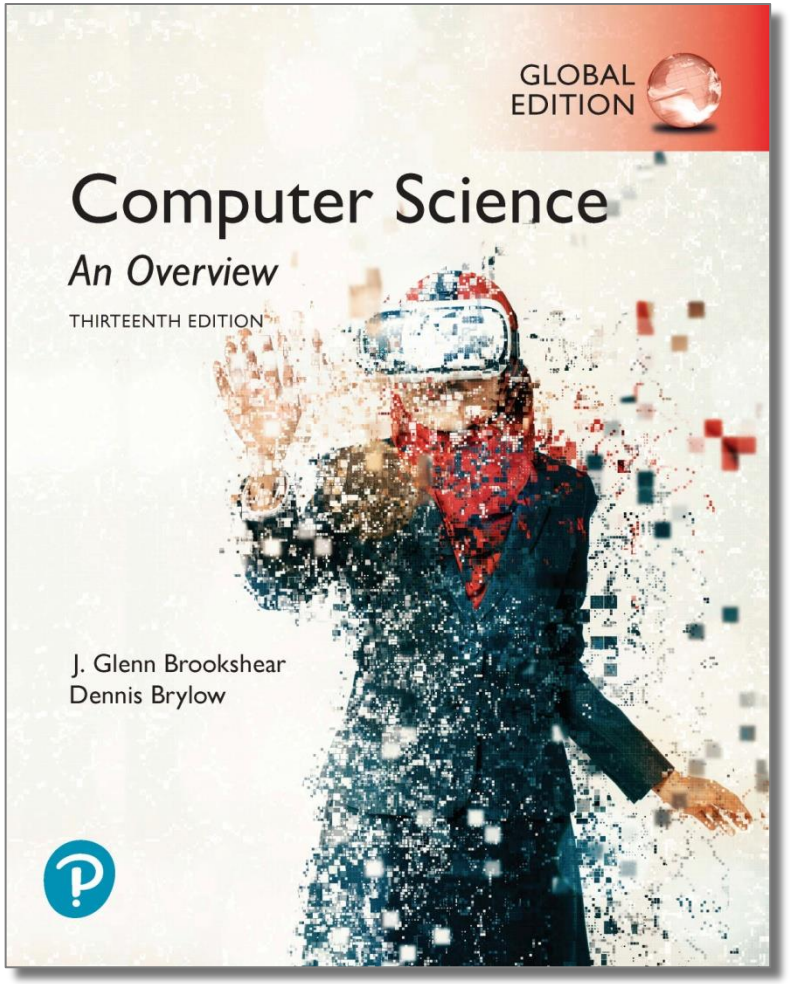
Computer Science An Overview

13th Edition, Global Edition



Chapter 3

Operating Systems

Chapter 3: Operating Systems

- 3.1 The History of Operating Systems
- 3.2 Operating System Architecture
- 3.3 Coordinating the Machine's Activities
- 3.4 Handling Competition Among Processes
- 3.5 Security

Examples of Operating Systems

- Windows
- UNIX
- Mac OS
- Solaris (Sun/Oracle machines)
- Linux

Smartphone Operating Systems

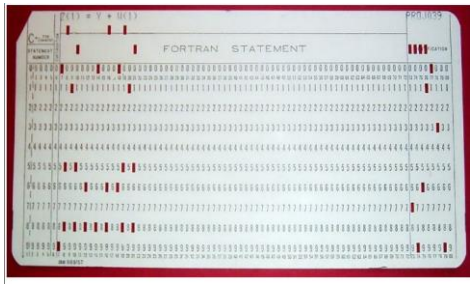
- Apple iOS
- Windows Phone
- BlackBerry OS
- Nokia Symbian OS
- Google Android

Functions of Operating Systems

- Oversee operation of computer
- Store and retrieve files
- Provide the user interface to request execution of programs
- Coordinate the execution of programs

3.1 History of Operating Systems

- Each program is called a “job”
- Early computers required significant setup time
- Each “job” required its own setup
- Operating Systems began as systems for simplifying setup and transitions between jobs



3.1 History of Operating Systems

- Batch processing (job queue)
- Interactive processing (real time)
- Time-sharing (one machine, many users)
- Multitasking (one user, many tasks)
- Multiprocessor machines (load balancing)
- Embedded Systems (specific devices)

Figure 3.1 Batch processing

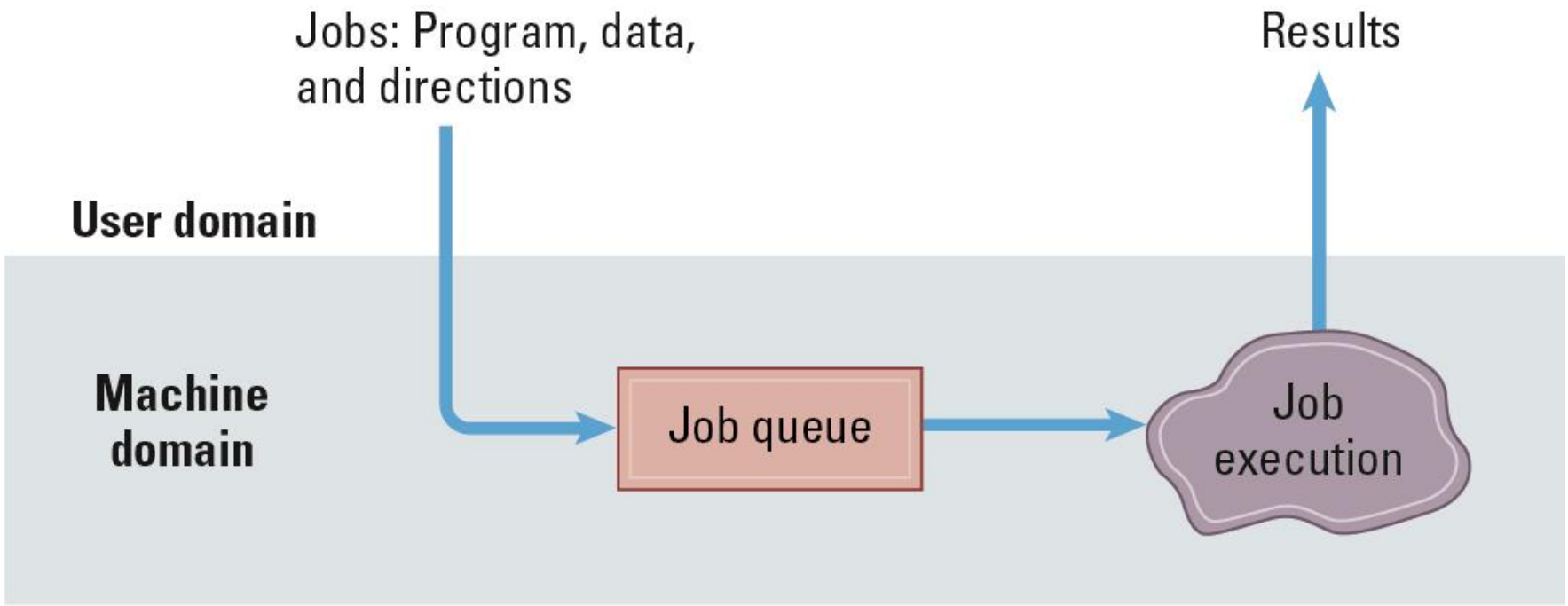
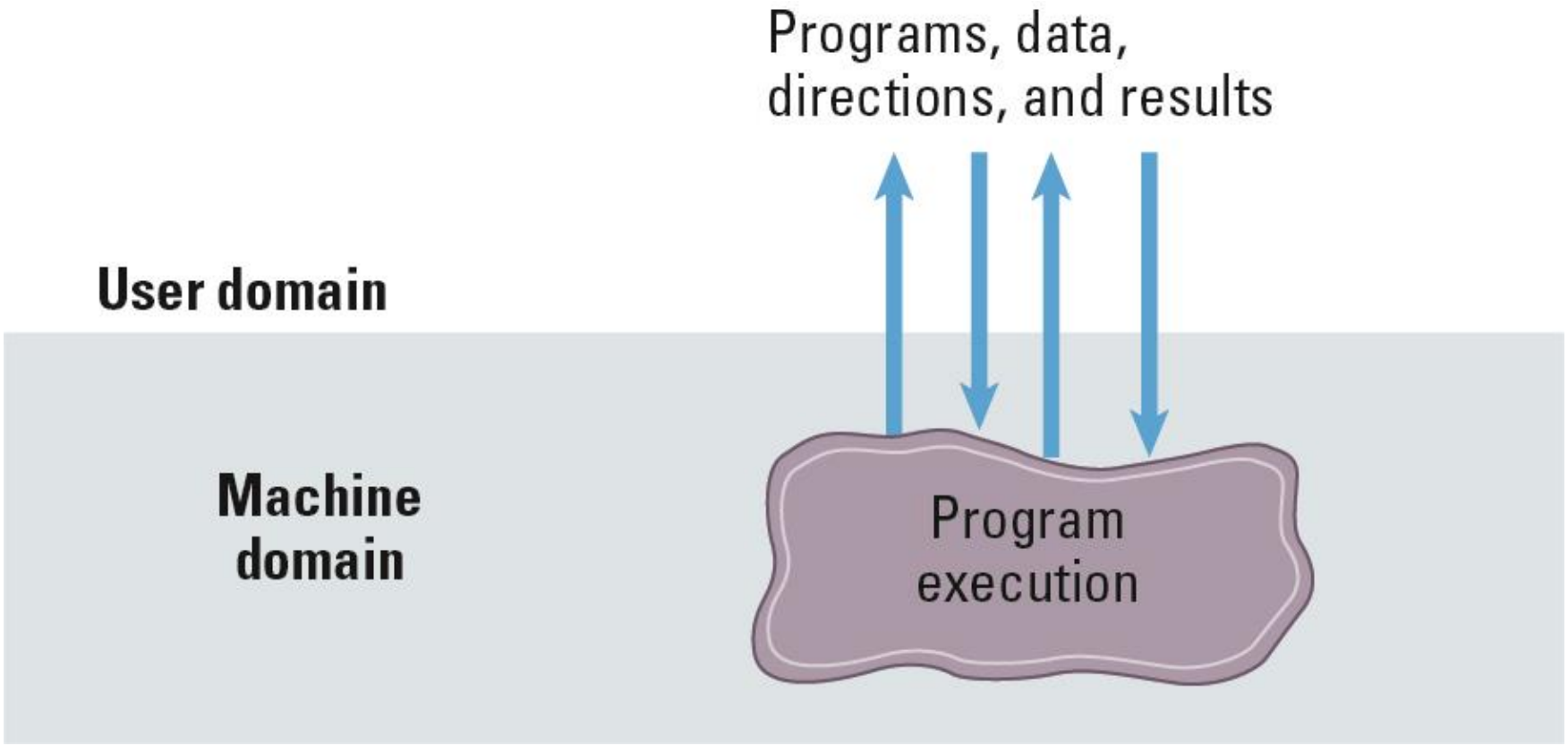


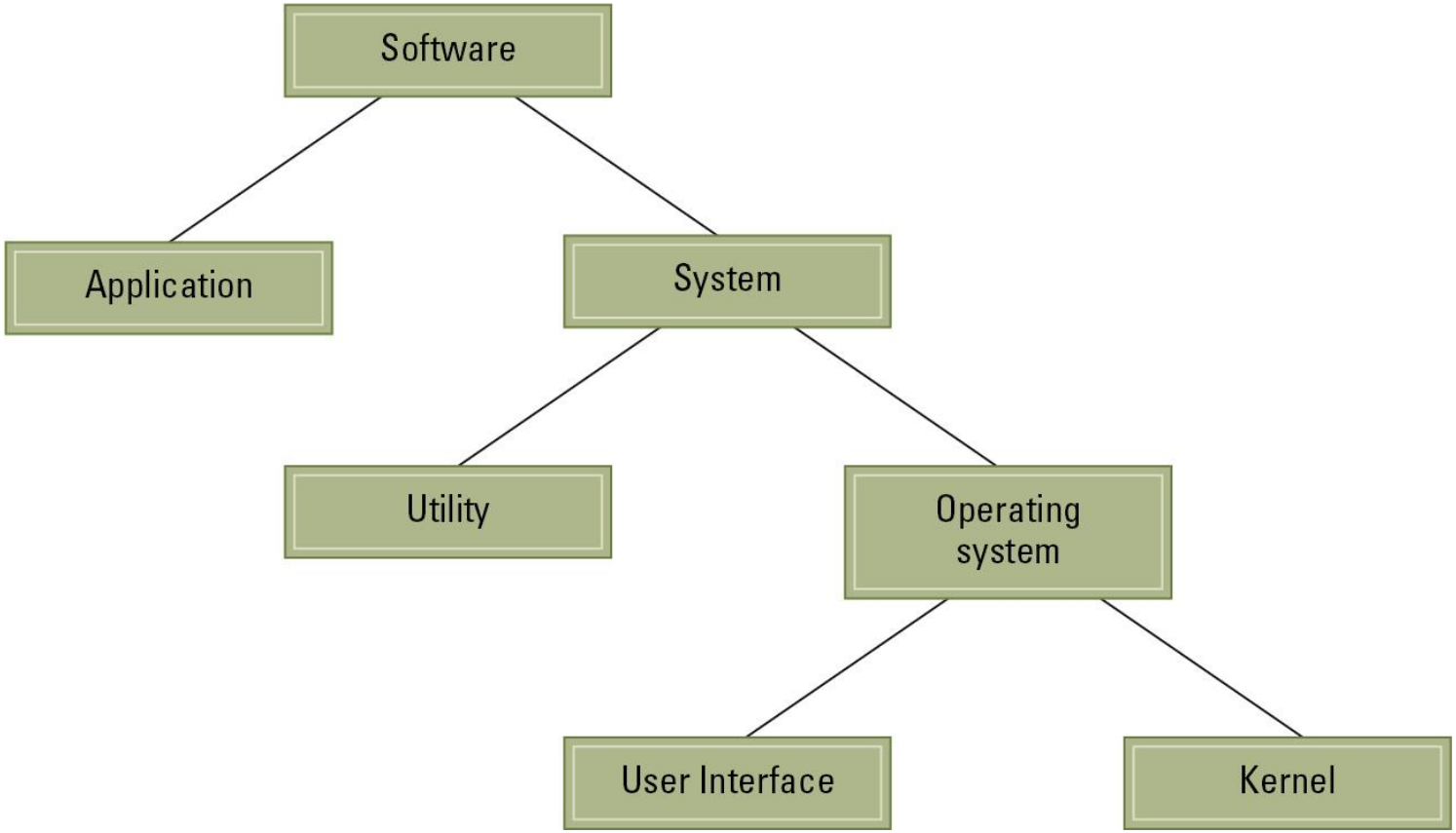
Figure 3.2 Interactive processing



3.2 Operating System Architecture

- Application software
  - Performs specific tasks for users (productivity, games, software development)
- System software
  - Provides infrastructure for application software
  - Consists of operating system and utility software

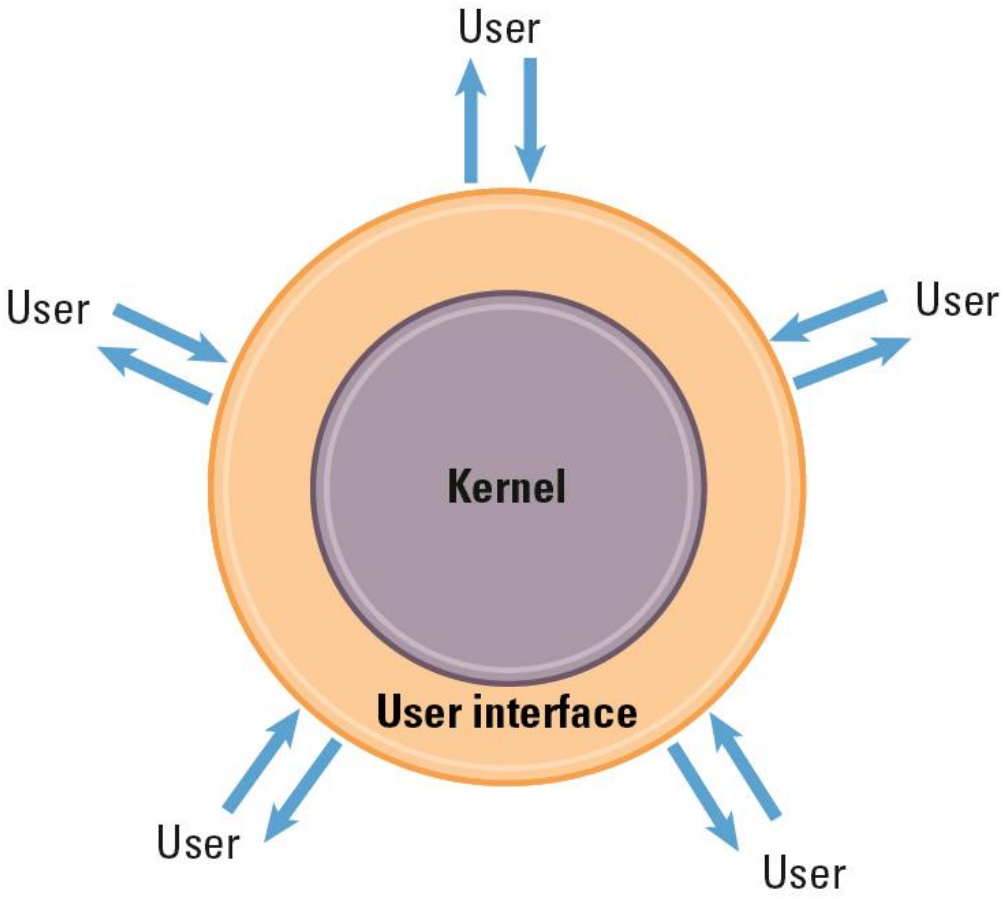
Figure 3.3 Software classification



Operating System Components

- **User Interface:** Communicates with users
  - Text based (Shell)
  - Graphical user interface (GUI)
- **Kernel:** Performs basic required functions
  - File manager
  - Device drivers
  - Memory manager
  - Scheduler and dispatcher

Figure 3.4 The user interface acts as an intermediary between users and the operating system's kernel



File Manager

- **Directory** (or **Folder**): A user-created bundle of files and other directories (subdirectories)
- **Directory Path:** A sequence of directories within directories

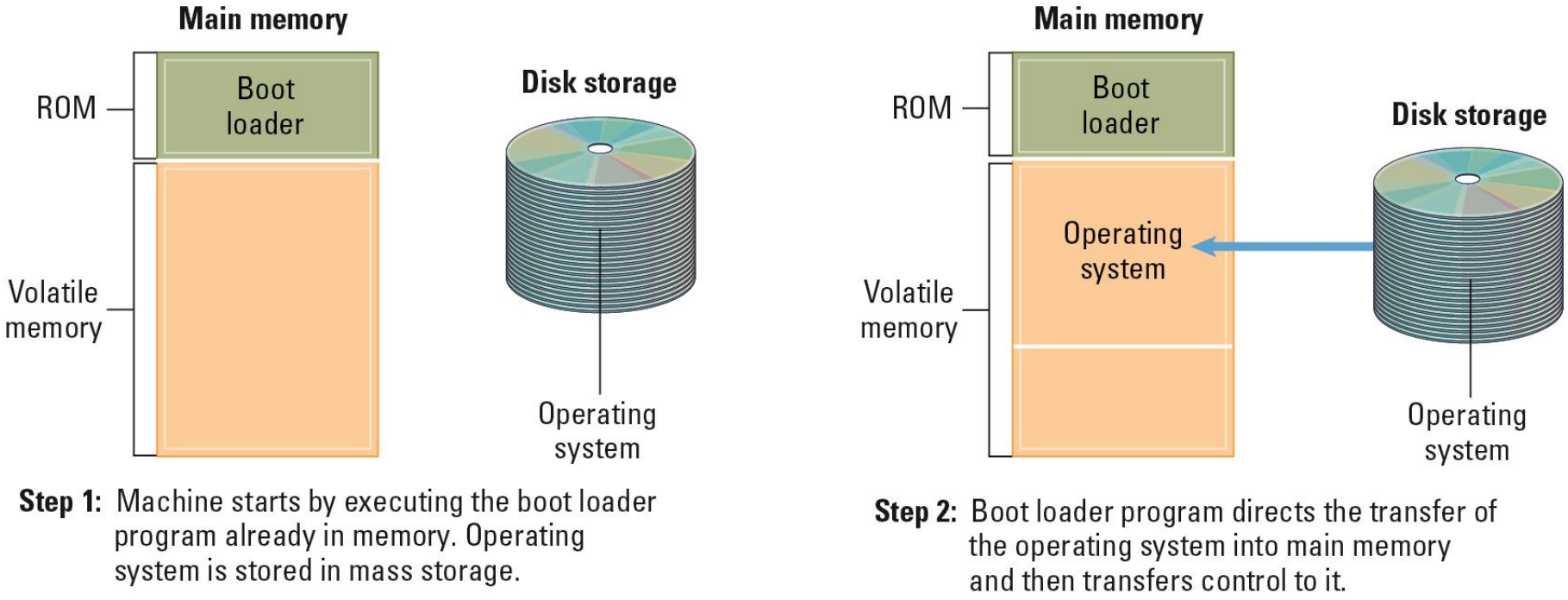
Memory Manager

- Allocates space in main memory
- May create the illusion that the machine has more memory than it actually does (**virtual memory**) by playing a “shell game” in which blocks of data (**pages**) are shifted back and forth between main memory and mass storage

Getting it Started (Bootstrapping)

- **Boot loader:** Program in ROM (example of firmware)
  - Run by the CPU when power is turned on
  - Transfers operating system from mass storage to main memory
  - Executes jump to operating system

Figure 3.5 The booting process



3.3 Coordinating the Machine's Activities

An operating system coordinates the execution of application software, utility software, and units within the operating system itself.

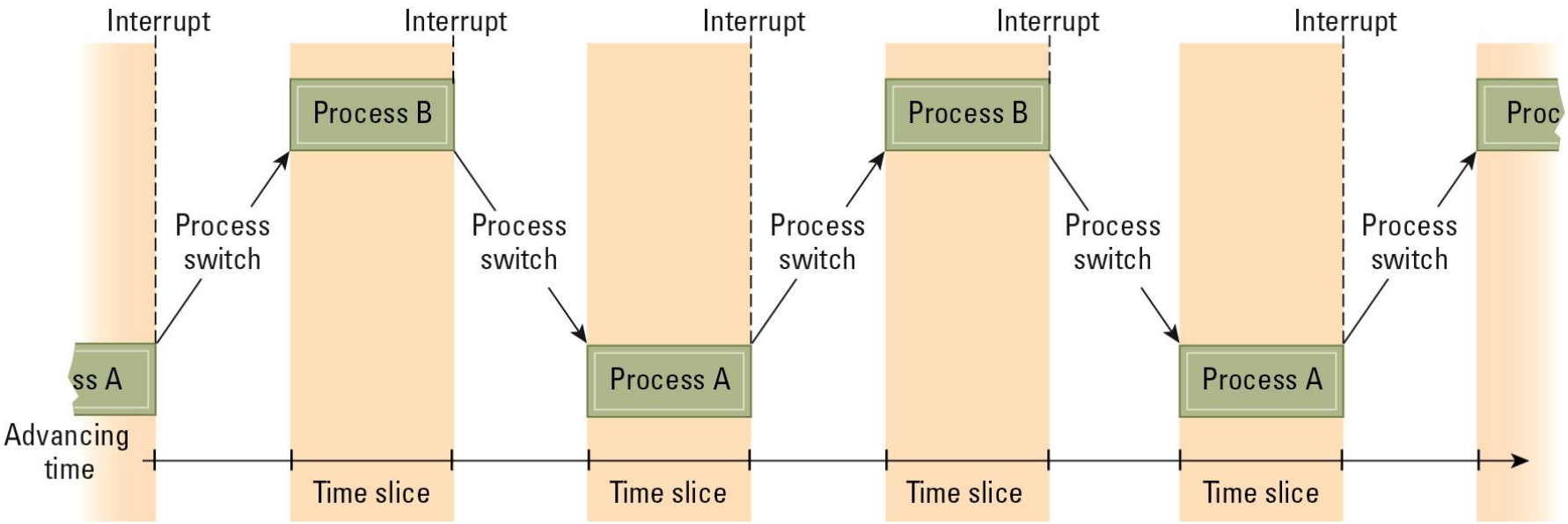
The Concept of a Process

- **Process:** The activity of executing a program
- **Process State:** Current status of the activity
  - Program counter
  - General purpose registers
  - Related portion of main memory

Process Administration

- **Scheduler:** Adds new processes to the process table and removes completed processes from the process table
- **Dispatcher:** Controls the allocation of time slices to the processes in the process table
  - The end of a time slice is signaled by an interrupt.

Figure 3.6 Multiprogramming between process A and process B

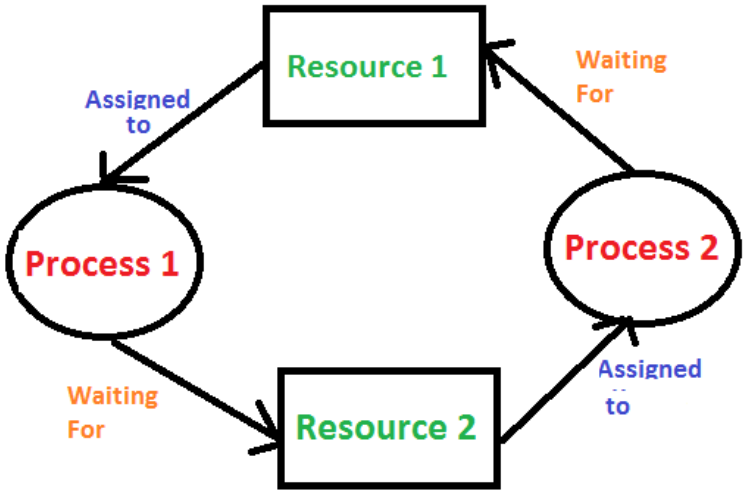


3.4 Handling Competition Among Processes

- **Semaphore:** A “control flag”
- **Critical Region:** A group of instructions that should be executed by only one process at a time
- **Mutual exclusion:** Requirement that only one process at a time be allowed to execute a Critical Region

Deadlock

- Processes block each other from continuing because each is waiting for a resource that is allocated to another
- Conditions required for deadlock
  1. Competition for non-sharable resources
  2. Resources requested on a partial basis
  3. An allocated resource can not be forcibly retrieved



Deadlock

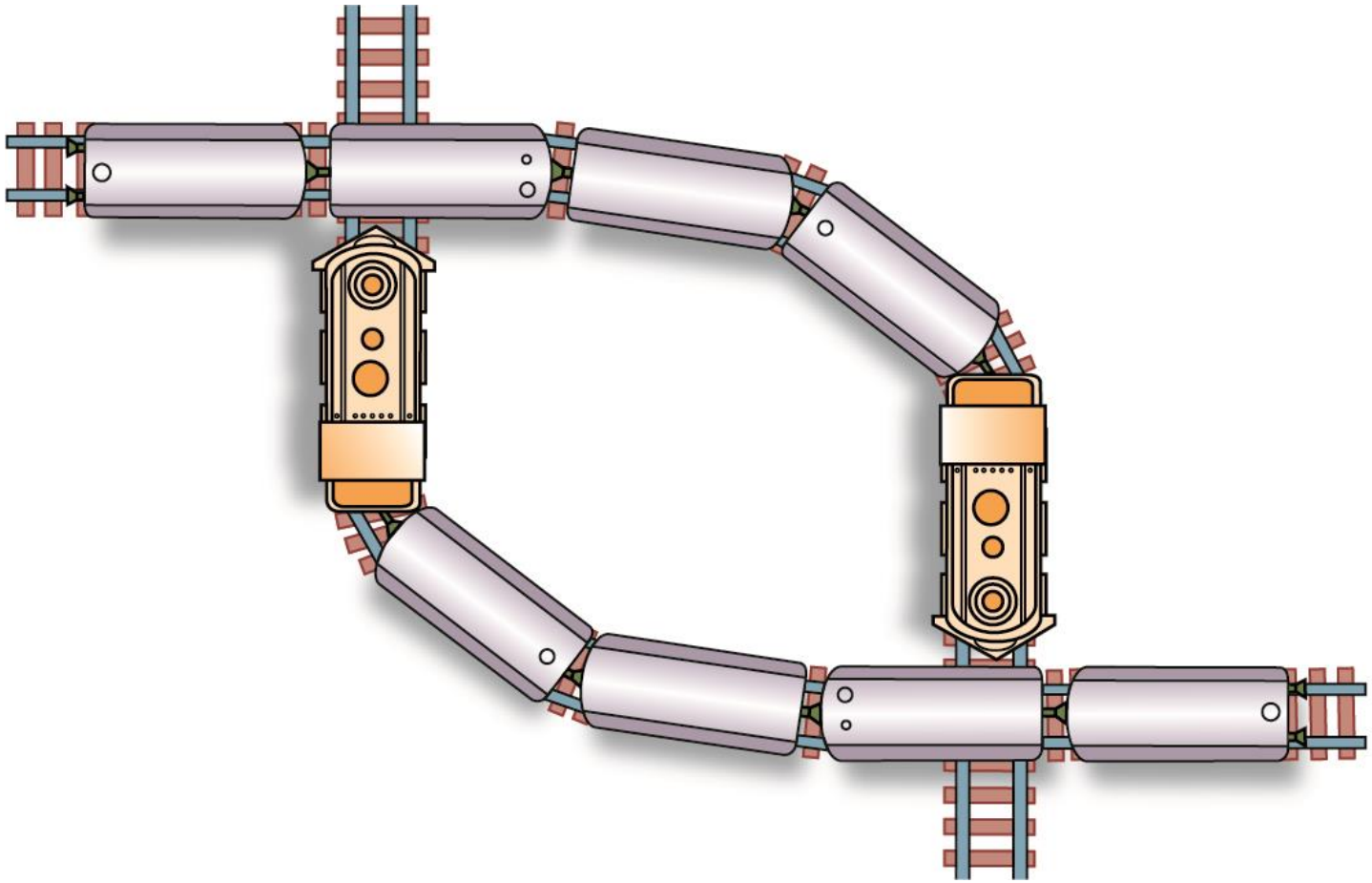
- 一般而言，我們可以處理死結問題(deadlock problem)使用下列三個方式其中之一

1. 我們可以使用一個協議(protocol)去預防或是避免死結(deadlocks)，確定系統永遠不會進入死結狀態(deadlocked state)。
2. 我們可以允許系統進入死結狀態(deadlocked state)，然後偵測它，恢復它。
3. 我們可以完全無視這些問題，假裝這些問題從來不曾發生過。

Deadlock

1. 互斥（mutual exclusion）：資源只能同時分配給一個行程，無法多個行程共享。
   - 對不可共用的資源類型而言，互斥一定成立，而可共用的資源類型，因為可以同時讀取相同檔案，所以一定不會產生。
2. 持有和等待（hold and wait）：一個行程可以在等待時持有系統資源。
   - 必須保證一個行程在要求一項資源時，不可以佔用任何其它的資源。
3. 禁止搶占（no preemption）：系統資源不能被強制從一個行程中登出。
   - 只要某個處理元要不到所要求的資源時，便把它已經擁有的資源釋放，然後再重新要求所要資源。
4. 循環等待（circular waiting）：一系列行程互相持有其他行程所需要的資源。
   - 確保循環式等候的條件不成立，我們對所有的資源型式強迫安排一個線性的順序。

Figure 3.7 A deadlock resulting from competition for nonshareable railroad intersections



3.5 Security

- Attacks from outside
  - Problems
    - Insecure passwords
    - Sniffing software
  - Counter measures
    - Auditing software

Security (continued)

- Attacks from within
  - Problem: A process that gains access to memory outside its designated area
  - Counter measures: Control process activities via privilege levels and privileged instructions